Learning with few examples the semantic description of novel human-inspired grasp strategies from RGB data

Lorenzo Collodi[1,3], Davide Bacciu[2], Matteo Bianchi[1,3], and Giuseppe Averta[4]

***Abstract*—Data-driven approaches and human inspiration are fundamental to endow robotic manipulators with advanced autonomous grasping capabilities. However, to capitalize upon these two pillars, several aspects need to be considered, which include the number of human examples used for training; the need for having in advance all the required information for classification (hardly feasible in unstructured environments); the trade-off between the task performance and the processing cost. In this paper, we propose a RGB-based pipeline that can identify the object to be grasped and guide the actual execution of the grasping primitive selected through a combination of Convolutional and Gated Graph Neural Networks. We consider a set of human-inspired grasp strategies, which are afforded by the geometrical properties of the objects and identified from a human grasping taxonomy, and propose to learn new grasping skills with only a few examples. We test our framework with a manipulator endowed with an under-actuated soft robotic hand. Even though we use only 2D information to reduce the footprint of the network, we achieve 90% of successful identifications of the most appropriate human-inspired grasping strategy over ten different classes, of which three were few-shot learned, outperforming an ideal model trained with all the classes, in sample-scarce conditions.**

I. Introduction

Endowing robotic systems with advanced grasping and manipulation capabilities still represents one of the grand challenges for the robotics community. Indeed, the functional pipeline for autonomous robotic grasping and manipulation [1] involves several elements, such as the object [2], the environment [3], the gripper [4], [5], and the task [6], which should be all considered while designing new mechanics, sensing and perceptual components, as well as planning and control strategies. An inspiration to successfully tackle these issues could arise from the human example. Indeed, human manipulation represents the golden standard, still unmatched, for robotic end effectors, in a world where objects are designed to be used by human hands [7]. Human inspiration has hence driven the development of mechanical systems that offer natural compliance, and deform with the environment to augment their grasping capabilities, like humans actually do [5], [3]. Furthermore, human inspiration can also provide informed guidelines for the planning phase, e.g. relying on the usage of learning methodologies [8] that can be used to deal with the difficulties in generating accurate models of the hand-object interaction, and adapt robot perception and control of unfamiliar objects [9]. Data-driven methods for robotic grasp generation are typically based on Neural Networks [10], [11], [12], [8] but usually suffer of some limitations, which are e.g. related to the quality and dimensionality of the data used for training [13], to the increase of the computational cost that grows with the complexity of the task and to the dimensionality of sensory information. It is also worth mentioning that most of the approaches proposed so far heavily rely on three-dimensional information (RGB-D, point clouds, etc.) [14], [15], [11], [2], which dramatically increases the complexity of the neural architecture w.r.t. 2D models, with negative impact on the deployment feasibility at the edge.

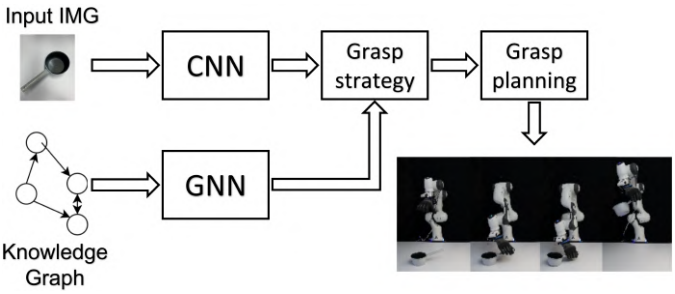

Fig. 1. General pipeline implemented in this work. From an input image, we first predict the best grasp from the taxonomy to lift the object, while integrating information from a knowledge graph. Then, given the images and the predicted grasps, a strategy to grasp the object is executed.

In addition, all the approaches proposed so far typically assume a pre-defined scenario. Yet, having robots in our lives requires them to interact with unstructured environments, where they may be not aware, at training time, of all the objects classes that are available in a realistic setting, for example because they are rare, or simply too costly. In this case, at the deployment time, the robot is made aware of a limited number of classes only, while other classes may appear only afterwards. To manage this issue, a simple yet trivial solution is to re-train from scratch the model when new data are provided, which is time consuming, inefficient and, potentially, even of negative impact for the knowledge learned beforehand. Alternatively, novel classes could be learned incrementally, preserving the knowledge already stored, and eventually using

Manuscript received: Sept 9, 2021; Revised: Dec 8, Year; Accepted: Jan 10, 2022. This paper was recommended for publication by Editor Markus Vincze upon evaluation of the Associate Editor and Reviewers' comments. This project has received funding from the EU Horizon 2020 research and innovation programme under Grants No. 871237 (Sophia) and 101017274 (Darko), and and by the Italian Ministry of Education and Research (MIUR) in the framework of the CrossLab project (Departments of Excellence). Additional support came from the PEGROGAM project (CHIST-ERA Call 2017 for Research Proposals).

[1] Centro di Ricerca "Enrico Piaggio", Universita' di Pisa, Largo Lucio Lazzarino 1, 56126 Pisa, Italy;[2] Dipartimento di Informatica, Universita' di Pisa, Largo B. Pontecorvo, 3, 56126 Pisa, Italy; [3] Dipartimento di Ingegneria dell'Informazione, Universita' di Pisa, Largo Lucio Lazzarino 1, 56126 Pisa, Italy; [3] Dipartimento di Automatica e Informatica, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10124 Torino, Italy giuseppe.averta@polito.itDigital Object Identifier (DOI): see top of this page

a small number of new examples. This setting, typically named few-shot learning, is a relatively new research field in Deep Learning (DL) [16], [17], [18], which provides a solution to the problem of supervised training sample size, one of the major limitations of modern DL models (see [19]). Indeed, especially in robotics - as for the task of grasping and manipulation - it is not uncommon the case that large datasets are not available, because of time and/or safety limitations [13] or because of the increasing deployment of robotic technologies in unstructured and unpredictable environments, which makes the creation of comprehensive datasets challenging. One solution to this problem is to collect data for a reduced number of classes (ideally the ones that are easier to experience), and then extend to others with few-shots. In addition, we rely on the fact that the generalization capabilities of the model to novel classes could be further enhanced by providing information associated to the relationship between the different grasp strategies (i.e. a similarity index between two leaves of a grasping taxonomy).

Therefore, in this paper we propose a contribution in this direction by developing a novel method for 2D vision driven grasping of common objects, which enables few-shot learning of novel grasp strategies. The usage of RGB-based information (as in [20], [21]) allows to reduce the cost of the sensing hardware and the network footprint. As briefly introduced before, this approach substantially differs from the literature, which usually heavily rely on 3D information [2], [14], [15], [11], [10], and represents - to the best of Authors' knowledge - the first example in literature of few-shot learning of human-inspired grasping strategies relying on 2D data only, opening interesting perspectives for the actual usage of these approaches directly at the edge.

More specifically, given a 2D image of a generic object, we aim at identifying the most appropriate human-inspired grasping strategy (see Fig. 1) through a combination of a Convolutional Neural Network (CNN), required to extract the features of the object, and a Gated Graph Neural Network (GGNN), which processes the information on the relationship between different grasping strategies. The neural architecture provides as output a semantic description of the grasping approach extracted from a human-inspired taxonomy [22]. The same image is then fed to a computer vision pipeline which, depending on the grasp predicted by our neural network, identifies a proper robotic implementation. We consider a set of human-inspired grasp strategies, which are afforded by the geometrical properties of the objects and identified from a human grasping taxonomy, and propose to learn new grasping skills with only a few examples. We extensively test our framework with a manipulator endowed with an under-actuated soft robotic hand (Fig. 2). Even though we use only 2D information to reduce the complexity of the network, we achieve 90% of successful identifications of the most appropriate human-inspired grasping strategy over ten different classes, of which three were few-shot learned, outperforming an ideal model trained with all the classes.

II. METHODS

Humans are capable of a great variety of grasping and manipulation strategies, which are typically described by taxonomies of grasp [22] and manipulation [23]. Each leaf of the taxonomy represents a specific geometric configuration of the hand, driven by the shape of the object and by the

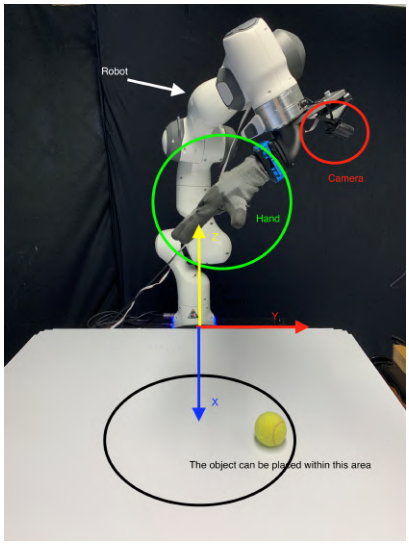

Fig. 2. Experimental Platform. A Franka Emika Panda is endowed with a soft end-effector and a webcam which looks at the area delimited by the black circle. Red, Blue and Yellow arrows stand for global X, Y and Z axes.

envisioned task. In addition, the variety of human behavior is further expanded considering the approaching strategy that precedes the execution of the grasp itself which - for some grasps typologies - could be more than one. For example, one could reach to grasp a bottle using a "large diameter" strategy (using the taxonomy proposed in [22]) with the palm facing downward if the bottle is lying down on the table, and with the palm in a vertical plane if the bottle is standing up. To replicate human strategies, it is possible to solve a classification problem over the leaves of a grasp taxonomy and associate to a given object the most appropriate human-inspired grasp that it would afford [7], assuming that these would result in a likely stabler grasp. However - considering the case in which robots are asked to work in unknown and unstructured environments - it may also occur that certain classes of objects are not available at training time [24] or they are available only with few samples. This typically results in unbalanced datasets, where some classes are sampled with an higher frequency. Training a deep neural architecture to perform classification over unbalanced classes is a difficult problem because it tends to introduce biases in the inference, resulting in poor predictive performance, especially for the minority classes [25]. Our solution to this problem is to implement a few-shot learning approach where we exploit the most represented K_{base} classes to train the model, and then extend the knowledge to the remaining K_{novel} classes relying on few examples. More specifically, our implementation moves from the architecture presented in [17] and is composed by two sub-models: a Convolutional Neural Network, which acts as feature extractor, and a Neural Network capable of processing graph data [26], specifically a Knowledge Graph Transfer Module (KGTM). To test our framework, we considered the Cornell Grasping dataset [11], which we manually labeled to associate each object with the most appropriate grasp strategy of the Feix et al. taxonomy [22]. To label the whole dataset we used ten leaves of the taxonomy, resulting in $K = 10$ classes. Because of the unbalanced distribution of grasps types in the Cornell Grasping dataset, we selected the three less-represented classes and used them as K_{novel}, while the remaining formed the K_{base} classes.

As a real test-bench, we considered a robotic platform composed by a Franka Emika Panda robot (https://www.franka.de/), endowed with a Pisa/IIT Softhand as End-Effector [5]. At the basis of the hand, we mounted a Logitech C922 Pro RGB camera, in an Eye-in-Hand configuration. During the experiments, the object was placed on a white background in front of the robot, so that the setting reproduces the one employed to collect the dataset used for the training of the network (see Fig. 2). The image also shows the coordinate frame referred to as "world frame" and the area where the object should be placed for the robot to be able to grasp it in the current configuration (radius $\approx 15[cm]$). Of note, this limit is manually imposed to avoid miss-classifications caused by image distortion, which may occur at the borders of the camera field of view (horizontal FOV $\approx 70^\circ$, vertical FOV $\approx 43^\circ$ in our case), but the working area can be arbitrarily expanded by changing the pose of the robot endpoint.

A. High Level Architecture

As briefly introduced in the previous section, our computational framework is composed by a Feature Extractor (FE) and a Knowledge Graph Transfer Module (KGTM). The interconnection between these is depicted in Fig. 3. More specifically, the input image is elaborated through the FE, a CNN that provides as output a vector of features which summarizes the information included in the image. Then, a Knowledge Graph is provided as input to the KGTM. The output of the KGTM is a vector of class embeddings that can be used to classify the image by finding the class with the lowest distance from the FE output in the embeddings space. The following sections will provide additional details on the implementation of each sub-element.

B. Feature Extractor

We used as a Feature Extractor the first 16 blocks of a pre-trained EfficientNet-B0 ([27]). The network was then fine-tuned for the intended task. To do this, we added at the end of the network a fully connected layer, which acts as a classifier, and further trained the Feature Extractor over the K_{base} classes. Then, the classifier output is given as input to a softmax layer to extract the confidence level of the prediction. Of note, the classification layer (fully connected) is needed for this phase only, and it is removed when the Feature Extractor is used in conjunction with the KGTM in the complete architecture, since its role is played in this case by the output of the KGTM.

C. Knowledge Graph

The Knowledge Graph given as input to the Knowledge Graph Transfer Module (see next subsection) provides a certain degree of a-priori knowledge on the relationship between different classes, defined in terms of kinematic distance. To quantify the distance between classes, we used the the HUST dataset (available at https://www.handcorpus.org/?p=1596 - last access Nov 15, 2021), a collection of kinematic recordings from subjects while performing the 33 grasps of the Feix taxonomy, and calculated the distance in the velocity domain between the kinematic synergies [28] associated to different leaves of the grasp taxonomy. We chose to use the velocity of the hand joints rather than joint angular values to increase the amount

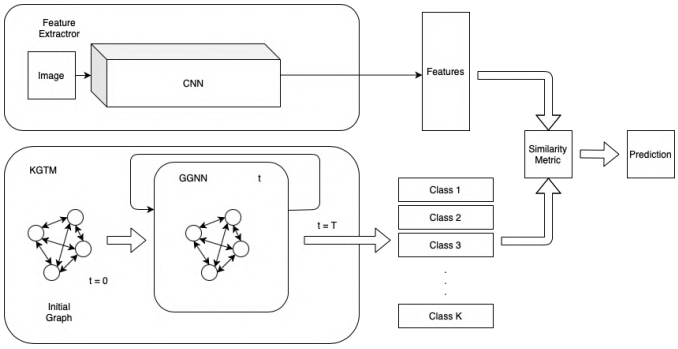

Fig. 3. General architecture of the KGTN, with its three main elements: the KGTM (Below), the Feature Extractor (Above), the Similarity metric employed to merge the two submodules (Right).

of information provided to the network. Indeed, we expected that relations associated to static hand shape are already mostly embedded in the information provided by the input image, while the analysis of joint angular velocities can inform on the dynamic evolution of the grasp. More specifically, this was achieved through the following steps:

1) We perform discrete differentiation of joint angles (16 Degrees of Freedom) to obtain joint velocities;
2) On these data, we apply PCA and extract the Principal Components (PCs) of the dataset;
3) We then extract the mean from all the entries of the dataset associated to each class, and project it in the subspace defined by the first six PCs, resulting in a PCs-based encodings of the average hand configuration (in velocity domain) associated to a given class. Of note, we used six components because these were sufficient to explain over the 86% of the total dataset variability;
4) We define a set of nodes $V = \{v_i | i \in [1,K]\}$. Each node represents one of the K classes and is associated to a label corresponding to the specific vector of activation of the synergies (extracted in the previous point);
5) The links between nodes $E = \{(v_i, v_j) | i \neq j\}$, instead, define a complete undirected graph, where each edge is associated to a 6-dimensional label $a_{ij}^k = |v_i^k - v_j^k|$ containing the absolute value of the difference of activation of the main synergies. In other terms, the larger is the distance in the PCs-domain between two different grasp types, the larger is the weight of the corresponding edge;
6) Finally, the node labels are reinitialized: the nodes corresponding the first K_{base} grasps are reassigned with a label equal to the corresponding row in the classifier trained when training the feature extractor. The remaining K_{novel} nodes are initialized randomly.

The output of this process is an undirected graph that encodes the information on the similarity between different grasp strategies in synergy space.

D. Knowledge Graph Transfer Module

The Knowledge Graph Transfer Module, in our implementation, is a Gated Graph Neural Network (GGNN), which takes as input a Knowledge Graph and outputs an array of class embeddings in the features domain (shared with the FE module). Of note, since the input knowledge graph is undirected, it is possible to identify cycles that prevent the update of weights in

one single parallelized calculation [29]. To overcome this issue, a common approach is to assume an iterative scheme, in which the state h_v^{t+1} of node v at iteration $t+1$ is updated using the states of the neighbors computed at the previous iteration t. Such an iterative scheme can be intended as a process that incrementally refines the representation of nodes.

In our implementation we have that for each time step $t < T$, where T is a hyperparameter of the module, and for each node of the graph, the GGNN updates the state of the node v with the following law:

$$h_v^{t+1} = \phi(h_v^t, \Psi(\{h_u^t, a_{uv} | u \in N_v\})), \tag{1}$$

where h_v^t is the node v state at time step t, N_v is its neighborhood and a_{uv} is the edge label associated to the edge between nodes u and v. The graph is assumed to be undirected and node states are initialized as $h_v^0 = v$, where v refers to both the node v and the label associated to the node.

The update law ϕ follows the dynamics of a Gated Recurrent Unit (GRU) [30], i.e.

$$h_v^{t+1} = \phi(h_v^t, A_v^t) = GRU((h_v^t, A_v^t)), \tag{2}$$

where

$$A_v^t = \Psi(\{h_u^t, a_{uv} | u \in N_v\}) = \sum_{u \in N_v} h_u^t E_G a_{uv} \tag{3}$$

and the GRU update law is implemented as in the following:

$$\begin{aligned} z_v^t &= \sigma(W_z A_v^t + U_z h_v^t) \\ r_v^t &= \sigma(W_r A_v^t + U_r h_v^t) \\ \widetilde{h}_v^t &= tanh(W A_v^t + U(r_v^t \odot h_v^t)) \\ h_v^{t+1} &= (1 - z_k^t) \odot h_v^t + z_k^t \odot \widetilde{h}_v^t. \end{aligned} \tag{4}$$

In Eq. 4, $tanh, \sigma$ stand for the hyperbolic tangent and the sigmoid function respectively, and $\odot$ represents the Hadamard product (i.e. an element-wise product). The term T is a hyperparameter (fixed to $T = 3$ in our implementation), while $W, W_r, W_z, U, U_r, U_z, E_G$ are trainable parameters. The output nodes are then evaluated as function of h_i^T and h_i^0, i.e. $v_i^O = f(h_i^T, h_i^0) = h_i^T$, and are used to classify the feature vectors extracted by the FE (see previous section). The particular selection of the metric depends on the task, in our implementation, given the output nodes $V = [v_1^O, v_2^O, ..., v_K^O]^T$, and the feature vector v_f, we defined $p = \sigma(V * v_f)$, where p is the vector of the probabilities assigned to each of the $\{K_{base} \cup K_{novel}\}$ and σ is the softmax activation function. The operator $*$ stands for the dot product between vectors. In other words, given a new object, for which we identify the feature vector v_f with the FE, we calculate the Euclidean distance between v_f and all the class embeddings V associated to $\{K_{base} \cup K_{novel}\}$, and select the closest one (which is the one with the highest dot product) as the output of the classification.

E. Ideal Model

To provide a ground truth for comparison, a standard CNN model was trained in an ideal setting in which the complete dataset is provided at training time, and the model is trained on all the classes together. Of note, this represents per-se the best choice, since in this way the model can converge to the optimal weights distribution for the given task. To produce a fair comparison, we manually tuned the hyperparameters of the Ideal Model for the same amount of time of our architecture, resulting in a model analogous to the one used for our Feature Extractor module, further extended with a fully connected layer, followed by a softmax, with $K = K_{base} + K_{novel}$ output classes.

F. Implementation of robotic grasping primitives

To increase the robustness of the inference, we mounted the camera directly at the end-effector level, and performed three different shots of the same object, taken from different perspectives (one frontal w.r.t. the manipulator, and two with a rotation around the vertical axis of approximately 35 degrees). The three views are taken with a camera-object distance of approximately $60[cm]$ and an inclination of the camera of $50°$ w.r.t. the horizontal plane. This comes with several advantages, since it partially compensates for uncertainties in calibration, it provides a simple way to estimate the pose of the object (described in the next section) and, most importantly, it can be used to improve the results of the network by ensembling. The pictures provided by the three views are used to perform inference, and the best grasp candidate is selected as the one with the highest average probability across the Network's prediction on the three views (averaging also ensures that the ensemble of the three predicted probability distributions is again a probability distribution, maintaining the normalization property). Then, the three images and the predicted grasp are used to plan a strategy for object grasping.

G. Object Segmentation and Localization

To identify the pose of the object to be grasped, we implemented a pose estimation procedure leveraging on the images taken from the three points of view. More specifically, we defined two subsequent procedures to identify the pose of the object, depending on the output of our classifier. For object associated to grasping primitives that could be executed only with a vertical approach (top-down) we identify only the centroid of the object. In case of grasp strategies that could be executed with different orientations of the hand (namely large diameter, parallel extension, small diameter and medium wrap), the estimation of object centroid is complemented with an estimation of its orientation. As a first step, from RGB images we performed background subtraction via OpenCV [31]. The image is converted to the HSV color space and then to a one-channel image by selecting the channel with the highest Standard Deviation. Median filtering and Canny algorithm [32] were applied to find the borders of the object and then closing operation compensates for errors in borders detection. A Suzuki algorithm [33] was then used to find the borders of the connected components. Subsequently, Flood filling is applied from the borders of the image. Among the remaining connected components, we identified the object by selecting the one closest to the center of the image apart from the largest one (which is assumed to be the background). Once the target object is extracted from the background, we perform Principal Component Analysis to extract the first principal component, which is used to identify the object orientation. The results of the whole segmentation procedure, plus the PCA output, are shown in Figure 4.

Given the segmented images, the lowest pixel on the vertical axis assigned to the object cluster ($p = [p_x, p_y, 1]^T$) is assumed

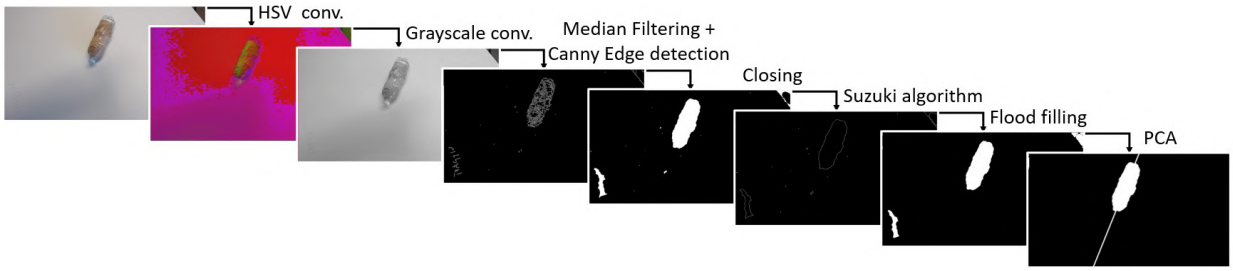

Fig. 4. Implementation of the segmentation pipeline: starting from the captured image (leftmost picture), the object is gradually extracted from the background by sequentially applying: HSV conversion, conversion to grayscale by selecting one channel, Median filtering and Canny algorithm, Closing operation, Suzuki algorithm and Flood filling. Finally, in the last image (rightmost), the object has been extracted from the background by selecting the connected component closest to the center of the scene and a PCA algorithm has been applied to find the main direction of the pixels distribution. The first Principal Component is visualized as a white line.

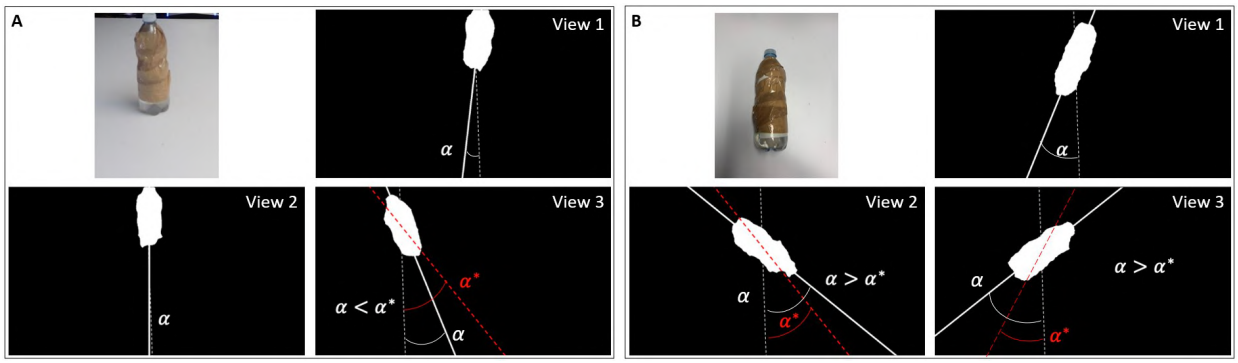

Fig. 5. A) Picture of a bottle standing up and resulting first PCs extracted from the three views. The angle between the direction of the PC and the vertical line is always below a threshold α^*; B) Picture of a bottle laying down on a table, and resulting first PCs extracted from the three views. The angle between the direction of the PC and the vertical line exceeds a threshold α^* for at least one of the three views.

to be on the table. The table height z_T is assumed to be known, so are the intrinsic camera matrix K and the camera-world transformation matrix T_c^w. The height of the table can be considered as a constraint to find the 3D position of the point ($p^w = [x, y, z, 1]^T$) using a pinhole model of the camera:

$$p^w = T_c^w \begin{bmatrix} kK^{-1}p \\ 1 \end{bmatrix} \tag{5}$$

and setting $p_3^w = z_T$ we can find k, which is the z coordinate of the point in camera frame. By repeating the whole procedure for the three views, it is possible to identify three 3D points, which are then averaged to estimate the center of the object face adjacent to the table p_{base}.

Then, only for the cases in which our neural architecture predicts a grasping strategy that could be executed with different orientation of the hand (i.e. for large diameter, parallel extension, small diameter and medium wrap only), the first PC of the three views is analyzed to estimate whether the object should be approached from above or from its side. More specifically, when an object is standing up on the table, requiring an approach from the side, the direction of the first PC extracted is similar between the three pictures and approximately vertical (see Fig. 5-A). Conversely, for objects laying down, at least one of the three pictures will yield a first PC direction significantly different than the vertical line (see Fig. 5-B). By manually setting a threshold on the angle between the vertical direction and the PCs direction we are able to discriminate between the two conditions (named α^* in Fig. 5). In our case, the angular threshold was manually tuned equal to $\alpha^* = 30°$. This procedure provides an estimation of the object 3D pose, and of its distribution (i.e. if the object is tall or short). At this point, we can plan and execute a consequent grasping primitive. More specifically, the actual execution is realized by the ROS package MoveIt! that implements trajectory planning with a Rapidly-exploring Random Tree (RRT) algorithm and executes the path with a joint velocity controller. The trajectory is executed in open loop, and end-effector force estimates are fed back to the controller to identify when the robot is touching the object (i.e. when the norm of the external force estimated by the robot exceeds a manually tuned threshold of 20 N) triggering the stop of the motion execution.

H. Grasp primitives

Since the end-effector we used (The Pisa/ITT SoftHand [5]) is capable of one single closure, the 10 classes defined above collapsed, for the grasp implementation phase, in four major grasping strategies, namely Vertical Approach (such as the one in Fig. 1), Power Horizontal Approach, Precision Horizontal Approach, and Sliding Grasp. Of these, the sliding grasp was not feasible because of limitations of the operative space of the robot. The Vertical Approach consists of grasps performed when an object requires a lateral grasp (vertical objects for which the classifier predicted one between large diameter, parallel extension, small diameter and medium wrap). For all the other cases, we implemented an horizontal grasping strategy, i.e. with the palm facing down. In those cases, for precision grasp strategies (i.e. tripod and palmar pinch) we implemented a Precision Horizontal Approach, in which we command the center of the tripod grasp (thumb, index and middle fingers) to reach the centroid of the object plus a vertical offset of $3[cm]$ before hand closure. Conversely, for power grasp strategies we implemented a Power Horizontal Approach, in which we command the base of the index finger

kinematic chain to reach the centroid of the object plus a vertical offset of $3[cm]$ before closure (we defined precision and power grasps as in [8]). It is worth remarking that this is not to be considered as a limitation of this work, but rather as a point of strength. Indeed, since in many cases there is a certain degree of ambiguity in the selection of the proper grasping approach (i.e. the same object may be grasped with different strategies) we can exploit the compliance embedded in the hand design and merge similar strategies (e.g. Large Diameter and Small Diameter) in the same grasping approach. At the same time, our neural architecture is still able to discriminate between all the classes. Therefore, in case of more dexterous hands, one could just expand the pool of grasping primitive available, while preserving our implementation.

III. Validation and Results

First, the network was trained and tested on a dataset built starting from the well known Cornell Grasp Dataset (available at https://www.kaggle.com/oneoneliu/cornell-grasp - last access Nov 15, 2021) and relabeling 642 images with the grasps from [22] that a human expert assigned with the highest probability of success. Then, both the network and the end-to-end framework were tested in a real setting as in Fig. 2.

A. Model training and validation

The dataset we used was first divided with a 70/30 stratified split between training set and test set. Then, 5 random stratified splits 60/40 with replacement were obtained from the training set, so as to obtain 5 different training/validation couples. These were used to perform cross-validation and select the best hyperparameters for model training. After cross-validation, the Feature Extractor and the Ideal Model were trained with the same parameters for 40 epochs with a SGD optimization algorithm. The tuned hyperparameters are learning rate and momentum. The same procedure was performed, using weights from the feature extractor pretrained on the 7 classes in K_{base}, to tune the hyperparameters and train the KGTM. Both the Ideal Model and the Feature Extractor are trained using a Stochastic Gradient Descent (SGD) optimization algorithm and using Focal Loss, a cost function introduced in [34] to overcome the problem of class imbalance. Focal Loss is characterized by a hyperparameter γ which, in our case, was set as $\gamma = 1$. The KGTM is trained with Focal Loss on all the classes with a 1:1 sampling ratio among the classes. As the dataset was very small in size, data augmentation was performed at training time with a random crop operator.

B. Dataset

Since we needed an application specific dataset, which was not available in literature, part of the images in the Cornell Grasp Dataset were relabeled in a way that was suitable for this problem. More specifically, from the original dataset we selected 642 images and assigned, to each of them, one of the classes of the Feix et al. taxonomy. All the entries of the dataset were covered by 10 different classes, namely Large Diameter, Parallel Extension, Small Diameter, Tripod, Medium Wrap, Palmar Pinch, Power Disk, Power Sphere, Sphere 4-finger and Extension. This resulted in a highly imbalanced dataset, in which 7 grasps were assigned to at least 40 images,

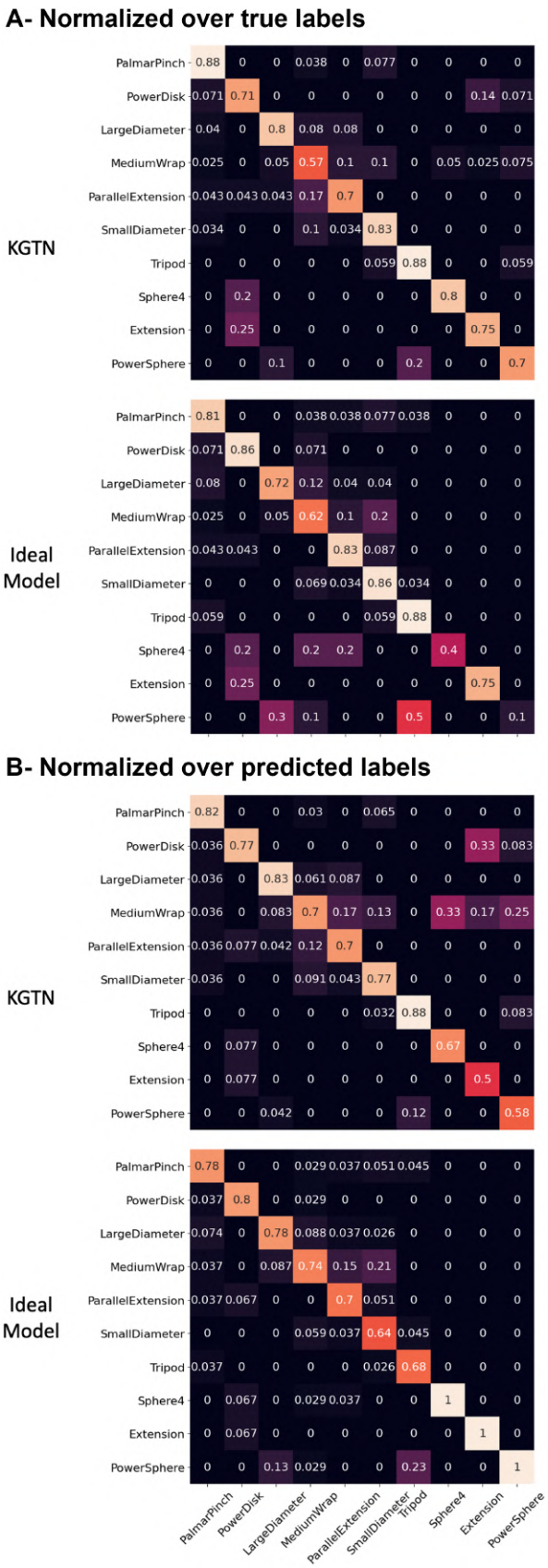

Fig. 6. Confusion matrices of both the KGTN (Above) and the Ideal Model (Below), normalized over true labels (sub-figure A, rows sum to 1) and predicted labels (sub-figure B, columns sum to 1). For each cell (t, p), the lighter is the color, the higher is the number of images with label t classified with the label p.

while the remaining three (i.e. Power Sphere, Sphere 4-finger and Extension) were assigned to less than 15 samples. For this reason, we selected the seven most represented classes to define the K_{base} classes on which to train the FE, and the remaining three were used as the K_{novel} subset of grasps on which to retrain the network through the KGTM.

C. Few-shot learning

After training the Ideal Model on all the classes and the KGTN on new classes starting with the weights trained on 7 classes, the models achieved an overall test accuracy of 73% and 75% respectively. The confusion matrices for the two models are shown in Figure 6. The rows correspond to the true labels, while columns to the predicted labels. The sub-figure A refers to values normalized w.r.t. true labels (i.e. it provides information on recall), while sub-figure B reports values normalized w.r.t. the predicted label (i.e. it provides information on precision). The grasps in K_{novel} correspond to the last 3 rows/columns.

D. Ablation

We also verified how a structural change in the knowledge graph encoding affects the model accuracy by testing different number of iterations T to update the nodes and by changing the definition of the knowledge graph. We observed that, while as expected with $T = 1$ (meaning that the status of the graph is updated in one single iteration) the training does not converge, performance increases with more iterations (75% for $T = 3$, 74% for $T = 5$). It is also worth noticing a small degradation of the accuracy while increasing T, which is related to the fact that each iteration updates the label of each node (i.e. the class embedding) depending on the neighborhood. While on one side this is the key mechanism that facilitates the generalization to novel classes, it also changes the class embeddings learned on the K_{base} classes, therefore still potentially forgetting a small amount of previous knowledge (albeit much less than the ideal model we used for comparison and other conventional CNN models as demonstrated in [17]). Therefore, it is important to carefully tune T as a trade-off between the two effects. Considering the type of information encoded in the knowledge graph, we tested an alternative approach in which the edges of the graph represent a distance in the synergistic space calculated over the joint angular values instead of velocities (hereinafter KGTM-Angle and KGTM-Velocity respectively). We observed that for KGTM-Angle the training still does not converge for $T = 1$, while performance is pretty similar for $T = 3$. However, for higher values of T we noticed that KGTM-Angle shows a larger drop in performances (72% instead of 74% of the velocity case). This suggests that the velocity encoding in the knowledge graph is more robust to forgetting than position encoding. Interestingly, we also observed differences in performance for specific classes. Indeed, KGTM-Velocity tends to be overall more accurate, while the KGTM-Angle provides higher accuracy only for small diameter, tripod e power sphere cases. One could speculate that this is likely caused by the fact that the objects shapes associated to those grasps tend to be significantly different from the other classes. In other words, there is a more strict relationship between the shape and the corresponding grasp than in the other classes. In this case, having a larger "static" information provided by the position encoding in the knowledge graph represents a point of strength. However, this is not true for all the other classes, in which the "dynamic" information provided by the velocity encoding in KGTM-Velocity yields better results. Another interesting result is that when the model predicts a wrong grasp, there is a tendency of the KGTM-Angle to assign more often a Power Disk graph w.r.t. the KGTM-Velocity. This could indicate a more evident bias of KGTM-Angle in that direction with respect to KGTM-Velocity.

Fig. 7. Set of 20 common-use objects employed to test our framework. The items varied in size, shape and color.

E. Grasping execution

The end-to-end model was then tested to assess its capability to grasp objects in a real setting. To do this, we applied our methods to a set of 20 different objects extracted from the YCB objects set and never used to train the architecture, which were presented to the camera three times in different configurations (see Fig. 7 and the companion video here: https://youtu.be/Tvl_5ni2j_Q). We recorded the grasps predicted by the Network to validate its performance in a real world setting. With the aim of testing the generalization and few-shot learning capabilities of the network, we evaluated the top-3 accuracy, reducing the bias of the evaluation procedure towards the more represented grasps [17]. With this, we assume that, if the network is able to recognize that the correct grasp is one of the most probable, it has also encoded the information about the image-grasp relation. The top-3 accuracy of the Network on the objects on which it was tested was 90%. Then, the success rate in grasping (calculated as an average across trials and objects) was 62%. However, there were two objects (the cup and the green bottle), that turned out to be too big to be grasped by one single hand and on which the success rate was 0%. Removing these two objects from the set, the success rate was 69%.The only grasp of K_{novel} that was tested in the real setting is the Power Sphere, on which the top-3 accuracy was 89% and so was the success rate in grasping.

IV. Discussions and Conclusions

The accuracy of the KGTN proved to be higher than the one of the Ideal Model. In addition, by looking at the class-specific results normalized over the true labels (6-A), we can observe that the Ideal Model is more often incorrect at classifying images from the classes in K_{novel}. Indeed, we can observe that the KGTN outperforms the Ideal Model on the new classes on which we tested few-shot learning. This is interesting because the K_{novel} classes are the ones less represented in the dataset, and, albeit using Focal Loss for training, the Ideal Model is not able to properly classify them. Our approach, instead, demonstrated the capability to overcome the issue related to class imbalance. Most of the confusion of the KGTN is between Power Disk/Extension and Medium Wrap/Sphere 4-finger, which were actually assigned to very similar objects and could

often be interchangeable for how the dataset was labeled. This highlights how the Knowledge Graph was successfully used to embed human knowledge about the image-grasp relation.

In addition, beyond the evident capabilities of KGTN to retain information related to previous training, it is also possible to verify that embedding new information about relationships also improves its performance on previously learned tasks, since KGTN tends to perform better on the K_{base} classes as well. This, on one side, highlights how the knowledge graph transfer approach was capable to successfully embed information from human knowledge and facilitate the task of few-shot learning of novel grasp strategies. On the other side, it also demonstrates that previous knowledge improves the classification capabilities even in cases of all data available at training time. For what concerns the actual execution of grasping, it is worth mentioning that the whole approach is independent from the hardware used for the implementation. Because of this, the believe that the achieved accuracy of 69% could be improved by modifying the pipeline in various points. In this sense, we shall mention the effects of using an underactuated hand. While the hand partially compensated for planning uncertainties in some cases, thus helping to successfully grasp the object, in other cases underactuation reduced the freedom to synthesize specific grasps of the taxonomy and therefore the grasps lacked the necessary stability and precision. To overcome these issues, we will test our framework with a more dexterous hand, targeting the synthesis of more articulated grasp primitives. Our future effort will be also devoted to extend of our pipeline including a model of object detection before our model. We expect that this would easily expand our method to multi-object scenarios, since the object detection architecture could identify a bounding box enclosing each object on the scene, on which our model is capable to classify the proper grasping strategy.

REFERENCES

[1] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, p. 289–309, Apr 2014.

[2] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1957–1964.

[3] C. Eppner, R. Deimel, J. Álvarez Ruiz, M. Maertens, and O. Brock, "Exploitation of environmental constraints in human and robotic grasping," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 1021–1038, 2015.

[4] R. Deimel and O. Brock, "Soft hands for reliable grasping strategies," in *Soft Robotics*. Springer, 2015, pp. 211–221.

[5] M. G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi, "Adaptive synergies for the design and control of the pisa/iit softhand," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 768–782, 2014.

[6] J. Bohg, K. Welke, B. León, M. Do, D. Song, W. Wohlkinger, M. Madry, A. Aldóma, M. Przybylski, T. Asfour *et al.*, "Task-based grasp adaptation on a humanoid robot," *IFAC Proceedings Volumes*, vol. 45, no. 22, pp. 779–786, 2012, 10th IFAC Symposium on Robot Control.

[7] J. J. Gibson, "The theory of affordances," in *The Ecological Approach to Visual Perception*. Houghton Mifflin Harcourt (HMH), Boston, 1977, ch. 8, pp. 127–137.

[8] C. D. Santina, V. Arapi, G. Averta, F. Damiani, G. Fiore, A. Settimi, M. G. Catalano, D. Bacciu, A. Bicchi, and M. Bianchi, "Learning from humans how to grasp: A data-driven architecture for autonomous grasping with anthropomorphic soft hands," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1533–1540, 2019.

[9] A. Billard and D. Kragic, "Trends and challenges in robot manipulation," *Science*, vol. 364, no. 6446, 2019.

[10] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 4461–4468.

[11] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[12] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 769–776.

[13] Y. Huang, M. Bianchi, M. Liarokapis, and Y. Sun, "Recent data sets on object manipulation: A survey," *Big data*, vol. 4, no. 4, pp. 197–216, 2016.

[14] Q. Lu, M. Van der Merwe, and T. Hermans, "Multi-fingered active grasp learning," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 8415–8422.

[15] J. Mahler, M. Matl, X. Liu, A. Li, D. Gealy, and K. Goldberg, "Dex-net 3.0: Computing robust vacuum suction grasp targets in point clouds using a new analytic model and deep learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5620–5627.

[16] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[17] R. Chen, T. Chen, X. Hui, H. Wu, G. Li, and L. Lin, "Knowledge graph transfer network for few-shot recognition," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, p. 10575–10582, Apr 2020.

[18] J. Chen, L. Pan, Z. Wei, X. Wang, C. Ngo, and T. Chua, "Zero-shot ingredient recognition by multi-relational graph convolutional network," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, pp. 10 542–10 550, 2020.

[19] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, "The computational limits of deep learning," *arXiv preprint arXiv:2007.05558*, 2020.

[20] Y. Xu, L. Wang, A. Yang, and L. Chen, "Graspcnn: Real-time grasp detection using a new oriented diameter circle representation," *IEEE Access*, vol. 7, pp. 159 322–159 331, 2019.

[21] S. Ainetter and F. Fraundorfer, "End-to-end trainable deep neural network for robotic grasp detection and semantic segmentation from rgb," *arXiv preprint arXiv:2107.05287*, 2021.

[22] T. Feix, J. Romero, H. Schmiedmayer, A. M. Dollar, and D. Kragic, "The GRASP Taxonomy of Human Grasp Types." *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 1–12, 2015.

[23] V. Arapi, C. Della Santina, G. Averta, A. Bicchi, and M. Bianchi, "Understanding human manipulation with the environment: a novel taxonomy for video labelling," *IEEE Robotics and Automation Letters*, 2021.

[24] C. Geng, S.-j. Huang, and S. Chen, "Recent advances in open set recognition: A survey," *IEEE transactions on pattern analysis and machine intelligence*, 2020.

[25] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[26] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Networks*, vol. 129, pp. 203–221, 2020.

[27] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[28] M. Santello, M. Flanders, and J. Soechting, "Postural hand synergies for tool use," *The Journal of neuroscience : the official journal of the Society for Neuroscience*, vol. 18, pp. 10 105–15, 12 1998.

[29] D. Bacciu, F. Errica, A. Micheli, and M. Podda, "A gentle introduction to deep learning for graphs," *Neural Networks*, 2020.

[30] K. Cho, B. v. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in *EMNLP*, 2014.

[31] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[32] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[33] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following," *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985.

[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.